(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 902 364 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
17.03.1999 Bulletin 1999/11

(51) Int. Cl.[6]: **G06F 9/445**, G06F 1/00

(21) Application number: **98116802.4**

(22) Date of filing: **04.09.1998**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **12.09.1997 JP 248745/97**

(71) Applicant: **Hitachi, Ltd.**
Chiyoda-ku, Tokyo 101-8010 (JP)

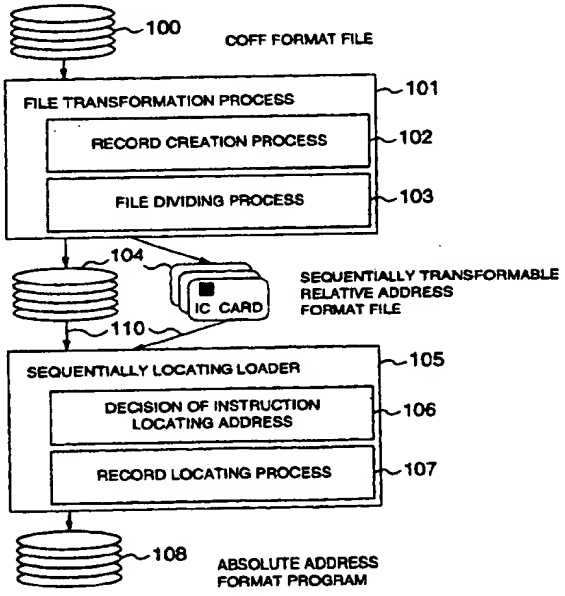
(72) Inventor: **Hirose, Takahiro**
Yokohama-shi (JP)

(74) Representative:
Strehl Schübel-Hopf & Partner
Maximilianstrasse 54
80538 München (DE)

(54) Method for loading a program

(57) A method of loading a program, wherein a relative address format file (100) is transformed into a program format including a set of records each having an instruction and instruction relocation information, and wherein a loader (105) receives the records one by one from a file of the above-mentioned program format, transforms the relative format program into an absolute address format program, and loads the absolute address format program on the memory.

FIG. 2



EP 0 902 364 A1

Description

BACKGROUND OF THE INVENTION

[0001] The present invention relates to a method for loading a program, and more particularly to a method for generating an absolute address program from a relative address program and loading the absolute address program into memory. In addition, the present invention relates to a computer software product, such as a computer readable medium that contains the above-mentioned method for loading a program.

[0002] As the object codes of programs, there are known absolute address codes and relative address codes. Absolute address codes are instructions that use absolute addresses when referring to data in memory or designating branch destinations. Programs that CPU can execute directly are only programs written using absolute addresses.

[0003] Relative address codes are instructions that designate memory addresses or branch destinations by using relative addresses such as symbolic addresses. Since CPU cannot execute a program which is written using relative addresses, a relative address program is converted into an absolute address program through a loader prior to execution by the CPU. Unlike the absolute address programs, it is possible to specify storage locations for relative address programs when they are loaded into memory. The relative address format programs are currently adopted in many computers because of the advantage that it is possible to freely select storage locations in memory for the relative address format programs.

[0004] A relative address format program file includes a program proper and relocation information. Relocation information is information that is referred to by the loader when converting a relative address format program into an absolute address format program. This is outlined in Chapter 5. Loader, particularly in Sub-chapter 5.1.5 Relocating Loader, pp. 170-174 of System Program I by John J. Donovan, 1976, published by Nippon Computer Kyokai. The most common of the program file formats is COFF (Common Object File Format). Its detailed description is given in Chapter 4. File Formats a. Out (4), pp. 1-2 of At&T UNIX SYSTEM V Programmers Reference Manual, 1990.

SUMMARY OF THE INVENTION

[0005] The loader for relocating a relative address format file reads the whole of a file, maps the file on a memory, and converts the file into an absolute address format file. A relative address format file, including relocation information, is often several times the size of an absolute address format file. Therefore, to enable the loader to operate, it is necessary to procure a working storage area several times larger than the whole of the program proper. If such a working storage area cannot be secured, the loader cannot load the program file, and the CPU cannot executes the program. It is the object of the present invention to provide a method for loading or executing a program even if a system involved does not have a memory resource sufficient enough to map a relative address format file in the memory.

[0006] To achieve the above object, the method for loading a program according to an aspect of the present invention comprises the steps of:

(1) converting a conventional relative address format file into a file of a program format having a set of records, each made up of an instruction and instruction relocation information;
(2) having the loader fetch records one by one from a file of a program format generated in step (1), transform the file into an absolute address format, and locate or dispose the file on the memory;
(3) when the file of the program format generated according to step (1) cannot be stored in a single medium, dividing the file into small files and storing them in a plurality of media; and
(4) having the loader sequentially load the small files, divided according to step (3), and locate them on the memory by rearranging them into a single program.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007]

Fig. 1 is a diagram showing a block diagram showing an example of a system to which the present invention is applied;
Fig. 2 is a diagram showing the procedure of a method of loading a program;
Fig. 3 is a diagram showing a relative address format file;
Fig. 4 is a diagram showing a sequentially transformable relative address format file in which records are created one record for each piece of relocation information.
Fig. 5 is a diagram showing an example of dividing a sequentially transformable relative address format file in which records are created one record for each piece of relocation information;
Fig. 6 is a flowchart showing a file transformation procedure for creating a sequentially transformable relative address format file in which records are created one record for each piece of relocation information or is a medium portion which stores therein a program for implementing the flowchart;
Fig. 7 is a flowchart showing a loader procedure for locating a sequentially transformable format file in which records are created one record for each piece of relocation information or is a medium portion which stores therein a program for implementing the flowchart;

Fig. 8 is a diagram showing a sequentially transformable relative address format file in which records are created one record for each instruction;

Fig. 9 is a diagram showing an example of division of a sequentially transformable relative address format file in which records are created one record for each instruction;

Fig. 10 is a flowchart showing a file transformation procedure for creating a sequentially transformable relative address format file in which records are created one record for each instruction or is a medium portion which stores therein a program for implementing the flowchart;

Fig. 11 is a flowchart showing a loader procedure for locating a sequentially transformable relative address format file in which records are created one record for each instruction or is a medium portion which stores therein a program for implementing the flowchart;

Fig. 12 is a diagram showing a memory image of executable codes with relocation information on the system;

Fig. 13 is a flowchart showing a procedure of loading a program into another system from the memory image of executable codes with relocation information or is a medium portion which stores therein a program for implementing the flowchart;

Fig. 14 is a diagram showing the outline of the method of loading a program, including steps of setting a condition for access permission and performing an encryption process; and

Fig. 15 is a block diagram showing a system configuration for loading an application program from a terminal device.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0008] Fig. 1 is a block diagram showing an example of a system to which the present invention is applied. Reference numeral 11 denotes an IC card terminal system. An IC card 19 is inserted into an IC card reader, not shown. This terminal system 11 includes a microcomputer chip 12, a flash memory 14, a card interface 15, a liquid crystal panel 16, and a key pad 17, and those component parts are interconnected by a common bus 18. The microcomputer chip 12 includes a RAM 13 as a working storage. The flash memory 14 contains an application program and data. The card interface 15 communications with the IC card. The liquid crystal panel 16 shows data such as the balance and menus. The key pad 17 accepts key inputs by the user. The user operates the keys to input instructions and data, and in response to the instructions, the terminal communicates with the IC card and executes a necessary process. The result of the process is displayed on the liquid crystal panel 16.

[0009] Fig. 2 is a diagram showing an example of process steps of the program loading method according to the present invention. Computers having an external storage device, such as a disk device or an IC card, carry out the program loading method according to the present invention. A file transformation process 101 accepts a relative address format file 100 as an input, and outputs a sequentially transformable relative address format file 104. A sequentially transformable relative address format file 104 can be stored in a disk, an IC card, or any other recording medium. If the file cannot be fully accommodated in a single medium, the file may be divided and stored in a plurality of media.

[0010] The most general one of relative address format files is COFF (Common Object File Format). The sequentially transformable relative address format file 104 is a file of a format which includes a set of records each made up of an instruction and instruction relocation information. The file transformation process 101 first creates records by a record creation process 102. There are two record creation methods: a method of creating records one record for each piece of relocation information and a method of creating records one record for each instruction. The file transformation process 101 collects the created records and outputs them to a file. If the file cannot be stored in a single medium, the file division process 103 divides the file into a plurality of operation units, in other words, files in this case. When an IC card is used to store a file, since its recording capacity is limited, it often becomes necessary to divide the file.

[0011] A sequentially relocating loader 105 reads a sequentially transformable relative address format file 104 (110), and outputs an absolute address format program. The absolute address format program can be directly located in memory and executed. The program may be written directly in memory or may be stored in a file. If there are a number of input files, the sequentially relocating loader performs a sequential loading process, thereby transforming those files into a single absolute address format program and outputting the program. The sequentially relocating loader 105 first reads the file header and decides instruction relocation addresses (106). On the basis of the addresses, the loader 105 executes a record relocation process (107), and generates an absolute address format program 108.

[0012] Fig. 3 shows an example of an object file of relative address format. This example is for purpose of illustration and shown in a simpler form than general COFF format files. A relative address format file 200 roughly consists of three major sections: a file header 201, a code block or field 202, and a relocation information block or field 207. Information written in the file header 201 is information on the attributes, the layout of areas and the like of the file. The code block 202 shows a string of instructions 203, 204, 205 and 206. Some instructions are those instructions that are subject to the relocation process and some others are not subject to

tr location process. Instructions 204 and 205 are no- elocation instructions. The instructions written in the code block are used as absolute address instructions as they are. Instructions 203 and 206 are to be subject to relocation and are transformed into absolute address instructions during loading according to relocation information 208 and 209.

[0013] There are two possible file formats of the sequentially transformable relative address format. Description will start with a method of creating records one record for each instruction and then a method of creating records one record for each piece of relocation information, is explained.

[0014] Fig. 4 shows an example of a sequentially transformable relative address format file 300, in which records are created one record for each piece of relocation information. A file 300 consists of a file header 301 and a record list 302. In the file header 301, file identification information and an offset for locating the record are written. The record list 302 has a plurality of records 303, 304 and so on arranged sequentially.

[0015] Records 303, 304 each include relocation information, a corresponding instruction to be relocated, and an instruction involving no relocation. Each record always includes a piece of relocation information and an instruction to be relocated. However, the record may not include an instruction involving no relocation or may include a plurality of instructions involving no relocation. The record 303 includes relocation information 208, an instruction to be relocated 203, and two instructions involving no relocation 204, 205. On the other hand, the record 304 includes relocation information 209 and an instruction to be relocated 206 but no instruction involving no relocation.

[0016] Because each record holds therein relocation information and an instruction to be relocated, the moment the loader reads a record, it can locate or dispose the record immediately.

[0017] Fig. 5 shows examples of dividing a file into sequentially transformable relative address format files 400, 403 in which records are created one record for each piece of relocation information. A sequentially transformable relative address format file can be divided into an arbitrary number of files, one file for each record list. The record list 302 in Fig. 4 is divided into a record list 402 and a record list 403, those lists 402 and 403 being respectively the components of a file 400 (file 1) and a file 403 (file 2).

[0018] Each file header includes file identification data and an offset in the code block. In a file header 401, however, only file identification information is written. Since the file 400 is the starting file, the instructions need to be located at the leading end of the code block. In a file header 404, file identification information and a code offset are written. In the file 403, which is the second file, the instructions need to be located in memory with a space for instructions of the start file left empty. From the header 404, it is understood that the instructions need to be located starting at a point 16 bytes from the leading end of the code block.

[0019] Fig. 6 shows a medium portion 101 that stores therein a program for implementing a procedure of the file transformation process for creating a sequentially transformable relative address format file in which records are created one record for each piece of relocation information. Steps 500 to 504 correspond to the record creation process 102 in Fig. 2. Steps 505 to 507 correspond to the file division process in Fig. 2. First, at Step 500, a relative address format file is read, and loaded into a flash memory 14. Then, at Step 501, a relocation information block 207 is inspected to check the number of entries of relocation information. Records of sequentially transformable relative address format as many as the number of entries are created. At Step 502, the entry in the relocation information field in each record is scanned, and relocation information is written in the record of the sequentially transformable relative address format.

[0020] Subsequently, the instructions in the code block 202 are written in the records. At this time, there is a difference in handling between the instructions to be relocated and the instructions involving no relocation. At Step 503, the instructions to be relocated are written in records one instruction in each record. At Step 504, the instructions involving no relocation, which are disposed after the written instruction to be located and before the next instruction to be relocated, are written in the same record where the immediately preceding instruction to be relocated has been written. Consequently, in some cases a plurality of instructions involving no relocation are written in a record or in other cases no instruction involving no relocation is written at all in a record.

[0021] At Step 505, the size of the resulting sequentially transformable relative address format file is estimated. Since the size of the file header is fixed, the size of the whole file can be estimated from the size of the storage area of the memory that the records occupy. If the size of the file is less than the storage capacity of the recording medium, the file as it is treated as a single file. When the file is larger than the storage capacity of the recording medium, the record list is divided somewhere into a plural number of files at Step 508. And, at Step 507, an instruction-offset value is calculated for each file, if necessary, and file headers are generated. Finally, at Step 508, records are written sequentially in the files.

[0022] For confidential programs, at Step 507, a condition for access permission are written in the file header. At Step 508, the records are fist encrypted and then written in a file. Records may be encrypted separately or the whole record list may be encrypted collectively. The general configuration for setting a condition for access permission or performing an encryption process will be described later by referring to Fig. 14.

[0023] Fig. 7 shows a medium portion 105 that stores therein a program for implementing a procedure of the

sequentially relocating loader for locating a sequentially transformable relative address format file in which records are created one record for each piece of relocation information. Steps 600 and 601 correspond to the instruction relocation address decision 106 in Fig. 2, and Steps 602 to 608 correspond to the record relocation process 107 in Fig. 2. At Step 600, a sequentially relocatable relative address format file header is read. At Step 601, the attributes of the file are referred to. If access permission information has been added to the file header, the access permission condition is checked, and if the permission condition is satisfied, the process continues. After this, a check is made to see if the file has been divided. If the file has been a single file, the process continues. If the file is found divided into a plurality of files, the instruction offsets of those files are checked. In the subsequent record relocation process 107, the instructions are relocated to locations that are separate by offset values from the reference point in the code block. This makes it possible to read the divided files into memory in arbitrary sequence.

[0024] At Step 602, the records are scanned sequentially. If the records have been encrypted, they are decrypted here. When all the records have been scanned, the record relocation process is completed. At Step 603, the relocation information of the record is fetched. Then, at Step 604, the instruction to be relocated is fetched. At Step 605, from this instruction and the relocation information that has been previously read, an absolute address instruction is generated, and the generated instruction is located or disposed in the code block in the memory. Then, at Step 606, a check is made to see if there is any instruction involving no relocation. If there is no instruction involving no relocation in the record, the process proceeds to the next record. However, if an instruction involving no relocation is included in the record, the instruction is written in the code block in the memory by performing Steps 607 and 608. After the instruction has been written in the memory, the process moves on to the next record.

[0025] After the process is finished, the relocation information in the code block in the memory is checked. If the file has been a single file, the instructions must have been relocated in all the code blocks. Notification that the program is ready for execution is given to the system. However, if a file was divided into smaller files, the instructions have not necessarily been relocated in all the code blocks. After it has been confirmed that all instructions were relocated, notification that the program is in ready condition is given to the managing program of the system.

[0026] A file 700 in Fig. 8 is an example of a sequentially transformable relative address format file in which records are created one record for each instruction. The file 700 includes a file header 701 and a record list 702. In the file header 701, file identification information and an offset for locating the record are written. The record list 702 is a list of records 703, 704, 705, 706 and so on arranged sequentially.

[0027] A record includes relocation information and an instruction. If the instruction is an instruction to be relocated, the record includes relocation information and an instruction. On the other hand, if the instruction is an instruction involving no relocation, the relocation information becomes null. In the records 703 and 706, because the instructions 203 and 206 are instructions to be relocated, they each have relocation information 208 and 209. In the records 704 and 705, because the instructions 204 and 205 are instructions involving no relocation, there is no relocation information for those instructions.

[0028] Each record holds therein information necessary for relocation. Therefore, the loader, on reading a record, can immediately locate or dispose the instruction.

[0029] Fig. 9 shows an example of division of a sequentially transformable relative address format file in Fig. 8 into files 800 and 803 in which records are created one record for each instruction. The sequentially transformable relative address format file can be divided into an arbitrary number of files, one file for each record list. The record list 702 in Fig. 8 is divided into a record list 802 and a record list 805 in Fig. 9, the record lists 802 and 805 being respectively the components of a file 800 (file 1) and a file 802 (file 2).

[0030] Each file header stores file identification information and a code offset in the code block. In a file header 801, however, only file identification information is written. Because the file 800 is the starting file, the instructions are to be located at the beginning of the code block. In a file header 804, file identification information and a code offset are written. Since the file 803 is the second file, when locating the instruction of the file 403, it is necessary to leave a space for the instructions of the first file. From the file header 804, it is understood that the instructions need to be located starting at a point 16 bytes from the leading end of the code block.

[0031] Fig. 10 shows a medium portion 101 that stores therein a program for implementing a procedure of the file transformation process for creating a sequentially transformable relative address format file in which records are created at the rate of one record for each instruction. Steps 900 to 904 correspond to a record creation process 102 in Fig. 2. Steps 905 to 908 correspond to a file division process 103 in Fig. 1. At Step 900, a relative address format file is read, and loaded into the flash memory 14. At Step 901, the code block is checked, the number of instructions is checked, and records as many as the instructions are created. At Step 902, an instruction is entered in the created record. At Step 903, the entry in the relocation information field is scanned to see if there is relocation information corresponding to the instruction. If there is not, it follows that instruction is an instruction involving no relocation. Therefore, the field for relocation information is to be left empty. If there is relocation information corresponding

to that instruction, it follows that the instruction is an instruction to be relocated. At Step 904, relocation information is written in the relocation information field of the record.

[0032] When all instructions have been written in the record, the size of a sequentially transformable relative address format file is estimated at Step 905. Since the size of the file header is fixed, the size of the whole file can be estimated by finding that storage size of memory which is occupied by the records. If the file size is within the storage size of a medium in which the file is written, the file is treated as a single file. If the file cannot be fully stored in the medium, however, the file is divided into a plurality of smaller files at Step 906. An instruction-offset value of each file is calculated, and file headers are created at Step 907. Finally, the records are written in the files at Step 908.

[0033] For confidential programs, at Step 907, a condition for access permission is written in the file header. At Step 908, the records are first encrypted and then written in the file. Records may be encrypted separately or the whole record list may be encrypted collectively. The general configuration for setting a condition for access permission or performing an encryption process will be described with reference to Fig. 1.

[0034] Fig. 11 shows a medium portion 105 that stores therein a program for implementing the procedure of the sequentially relocating loader for relocating a sequentially transformable relative address format file in which records are created one record for each instruction. Steps 1000 and 1001 correspond to the instruction relocation address decision 106 in Fig. 2, and Steps 1002 to 1007 correspond to the record relocation process 107 in Fig. 2. At Step 1000, the file header is read. At Step 1001, the attributes of the file are referred to. If access permission information is added in the file header, the access permission condition is checked. If the condition is satisfied, the process continues. They, the file is checked to see if it has been divided. If the file has been a single file, the process continues. If a file was divided into a plurality of smaller files, the instruction offsets of those files are checked. In the subsequent process 107, instructions are located at locations that are separate by offset values from the reference point in the code block.

[0035] Subsequently, at Step 1002, records are scanned sequentially. The records, if they have been encrypted, are decrypted here. When all records have been scanned, the process 107 is completed. At Step 1003, the record is inspected to see if relocation information has been written. If there is no relocation information, the instruction is written as it is in the code block at Step 1003. If there is relocation information, the relocation information is read at Step 1005, and then an instruction to be relocated is read at Step 1006. At Step 1007, from the relocation information and the instruction to be relocated, an instruction at an absolute address is generated, and the generated instruction is written in the code block in the memory.

[0036] After all records have gone through the above the process, the relocation information in the code block in the memory is checked. If the file has been a single file, the instructions must have been relocated to absolute addresses in all code blocks. Notification that the program is now in executable condition is given to the manager of the system. If a file was divided into smaller files, the instructions have not necessarily been relocated in all the code blocks. When it has been confirmed that all files were loaded and the instructions were relocated in all the code blocks, notification that the program is ready for execution is the manager of the system.

[0037] Fig. 12 is a diagram showing a memory image of executable codes with relocation information. The whole program has been transformed into an absolute address format ready for operation and loaded on the memory. Once the program has been transformed into an absolute address format, the relocation information is lost, so that the relocation process cannot be performed any more. In anticipation of future relocation, relocation information must be stored in another storage area.

[0038] An area 1101 shows a memory image on the system. An area 1103 denotes a code block of an executable application program. Areas 1104, 1105, 1106 and 1107 denote instructions in the code block. An area 1109 denotes a storage block for storing relocation information for the instructions. The instructions 208 and 209 are relocation information. An area 2220 denotes a storage block for storing additional information other than relocation information. The areas 1109 and 1110 are not required for execution of the instructions, but they are used when the program in the code block 1103 is relocated and loaded to another system.

[0039] An area 1102 is a system block, and an area 1111 is a block for storing another application program.

[0040] Fig. 13 shows a medium portion that stores therein a program for implementing a procedure for loading to another terminal a program from the memory image of executable codes added with relocation information shown in Fig. 12. At Step 1201, the code block in the memory of the sending terminal is checked. If there is not any instruction, the procedure is terminated. If there is an instruction, the instruction fetched at Step 1202, and the relocation information block is checked for relocation information corresponding to the instruction at Step 1203. If there is not relocation information, at Step 1205, the instruction is transferred to the terminal to which the program is loaded. If there is relocation information, at Step 1204, the relocation process is performed. After the process is finished, the instruction is transferred to the destination terminal at Step 1205'.

[0041] After the instruction has been transferred, the process proceeds to Step 1206. When necessary, a command is issued directing that the program, which has been loaded to the one other terminal, should be

loaded to yet another terminal. If a command to load the program to another terminal is not issued, the process is terminated. When the program is loaded to yet another terminal, relocation information is required. At Step 1207, relocation information is transferred to the terminal as the loading destination.

[0042] Fig. 14 shows a general configuration of a procedure including the processes of setting a condition for access permission and performing an encryption process. These processes are preceded by a common basic procedure. After main steps of file transformation are finished, a permission condition writing process 1401 is first conducted, and followed by an encryption process 1402.

[0043] If a permission condition has been set or encryption has been done in a sequentially transformable relative address format file 104, other processes must precede the main relocation process. More specifically, a decryption process 1403 is executed to make the contents of the file readable. Then, a permission condition reading process 1404 is executed to confirm the permission condition. If the permission condition is satisfied, the process continues. If not, the process is terminated. From the sequentially relocating loader on, the same basic procedure is performed.

[0044] Fig. 15 shows an example of a system configuration for loading an application program from a terminal by applying the procedure in Fig. 13. Both terminals 1501 and 1502 are identical with the earlier-described terminal 11. In this example, the application program is loaded from the terminal 1501 to the terminal 1502. A flash memory 14 in the terminal 1501 contains an application program of a format of the memory image for executable codes accompanied by relocation information in Fig. 12. A microcomputer 12 in the terminal 1501 executes the loading procedure shown in Fig. 13 to generate an application program that can be loaded directly. The terminal 1502 receives the application program from the terminal 1501 and writes it into its own flash memory 14.

[0045] There are many methods for connecting terminals. In the example in Fig. 15, the slots for IC cards are utilized for communication between terminals. Communication connectors for loading may be provided separately. Application programs can be loaded by infrared communication or radio communication.

[0046] The above-mentioned processes and steps according to the present invention can be embodied in computer software programs, including computer readable media, such as a CD-ROM and a floppy disk.

[0047] According to the above-mentioned embodiment, a program can be loaded or executed in a system without memory resources sufficient to load a relative address format file into the memory (memory 13 in Figs. 1 and 15).

Claims

1. A method of loading a program comprising the steps of:

   reading a sequentially transformable relative address format program, including a set of records each having an instruction and instruction relocation information corresponding to said instruction (110); and
   sequentially relocating instructions by reading said records one by one to thereby generate an absolute address format program (602 - 608, 1002 - 1007).

2. A method according to Claim 1, further comprising the step of mapping said absolute address format program, which has been generated, on the memory (605, 1007).

3. A method of loading a program comprising the steps of:

   transforming a first relative address format program including an instruction string and relocation information into a second relative address format program including a set of records each having an instruction and instruction relocation information corresponding to said instruction (101); and
   sequentially relocating instructions by reading said records one by one to thereby generate an absolute address format program (602 - 608, 1002 - 1007).

4. A method of loading a program comprising the steps of:

   transforming a first relative address format program including an instruction string and relocation information into a second relative address format program including a set of records each having an instruction and instruction relocation information corresponding to said instruction (101); and dividing said second relative address format program into a plurality of storage units (400, 403; 103); and
   storing each said storage unit in each storage medium (103).

5. A method according to Claim 4, further comprising the step of sequentially relocating instructions by reading said records one by one to thereby generate an absolute address format program.

6. A method according to Claim 5, further comprising the step of mapping said absolute address format program, which has been generated, on the mem-

ory.

7. A method of loading a program comprising the steps of:

transforming a first address format program including an instruction string and relocation information into a second relative address format program including a set of records each having an instruction and instruction relocation information corresponding to said instruction (101);
dividing said second relative address format program into a plurality of storage units (400, 403; 103); and providing each said storage unit with a header having set therein relocation information for each said storage unit (103); and
loading said storage units on the memory by reading said storage units in an optional sequence; and sequentially relocating said instructions by reading said records one by one according to relocation information in the header of each said storage unit that has been read in.

8. A method of loading a program comprising the steps of:

reading a sequentially transformable relative address format program, including a set of records each having an instruction and instruction relocation information corresponding to said instruction (110);
encrypting said program that has been read in (1402);
decrypting said program while sequentially relocating said instructions by reading said records one by one to thereby generate an absolute address format program (602 - 608, 1002 - 1007).

9. A method according to Claim 3, further comprising the step of storing a header having a permission condition set therein into said second relative address format program (1401), wherein when said permission condition is satisfied, instructions are relocated sequentially by reading said records one by one.

10. A method according to Claim 3, further comprising the step of storing a header having attributes information set therein into said second relative address format program (1401).

11. A method of loading a program comprising the steps of:

transforming a first relative address format program including an instruction string and relocation information into a second relative address format program including a set of records (302) having a group of instructions including instruction relocation information and at least one instruction corresponding to said relocation information (101); and
sequentially relocating instructions by reading said records one by one to thereby generate an absolute address format program (602 - 608, 1002 - 1007).

12. A method according to Claim 3, further comprising the step of storing said transformed program in at least one IC card, wherein said program is loaded in memory by generating an absolute address format program by sequentially relocating the instructions by reading said records one by one from said IC card.

13. A computer software product including a computer readable medium, said medium having stored thereon:

a first code section for holding a sequentially transformable relative address format program including a set of records each having an instruction and instruction relocation information corresponding to said instruction, and for having said relative address format program read into a computer (110); and
a second code section for generating an absolute address format program by sequentially relocating the instructions by reading said records one by one (602 - 608, 1002 - 1007).

14. A computer software product according to Claim 13, said medium having further stored thereon:

a third code section for loading said absolute address format program, which has been generated, on the memory (605, 1007).

15. A computer software product including a computer readable medium, said medium having stored thereon:

a code section for transforming a first address format program including an instruction string and relocation information into a second relative address format program including a set of records each having an instruction and instruction relocation information corresponding to said instruction (101); and
a code section for generating an absolute address format program by sequentially relocating instructions by reading said records one

by one (602 - 608, 1002 - 1007).

16. A computer software product including a computer readable medium, said medium having stored thereon:

a code section for transforming a first address format program including an instruction string and relocation information into a second relative address format program including a set of records each having an instruction and instruction relocation information corresponding to said instruction (101); and
a code section for dividing said second relative address format program into a plurality of storage units (400, 403; 103), and storing said storage units into a plurality of corresponding recording media (103).

17. A computer software product according to Claim 16, said medium having further stored thereon:

a code section for generating an absolute address program by sequentially relocating instructions by reading said records one by one in each said storage unit.

18. A computer software product according to Claim 17, said medium having further stored thereon:

a code section for loading said absolute address format program, which has been generated, on the memory.

19. A computer software product including a computer readable medium, said medium having stored thereon:

a code section for transforming a first relative address format program including an instruction string and relocation information into a second relative address format program including a set of records each having an instruction and instruction relocation information corresponding to said instruction (101); and
a code section for dividing said second relative address format program into a plurality of storage units (400, 403; 103); and providing each said storage unit with a header having set therein relocation information of each said storage unit (103); and
a code section for loading said storage units on the memory by reading said storage units in an optional sequence, and sequentially relocating said instructions by reading said records one by one according to relocation information in the header of each said storage unit that has been read in (602 - 608, 1002 - 1007).

20. A computer software product including a computer readable medium, said medium having stored thereon:

a code section for holding a sequentially transformable relative address format program including a set of records each having an instruction and instruction relocation information corresponding to said instruction, and for having said relative address format program read into a computer (110);
a code section for encrypting said program that has been read in (1402); and
a code section for decrypting said program while sequentially relocating said instructions by reading said records one by one to thereby generate an absolute address format program (602 - 608, 1002 - 1007).

21. A computer software product according to Claim 15, said medium having further stored thereon a code section for storing a header having a permission condition set therein into said second relative address format program, wherein when said permission condition is satisfied, said instructions are relocated sequentially by reading said records one by one.

22. A computer software product according to Claim 15, said medium having further stored thereon a code section (1401) for storing a header having attributes information set therein in said second relative address format program.

23. A computer software product according to Claim 15, said medium having further stored thereon:

a code section for transforming a first relative address format program including an instruction string and relocation information into a second relative address format program including a set of records (302) having a group of instructions including instruction relocation information and at least one instruction corresponding to said relocation information (101); and
a code section for generating an absolute address format program by sequentially relocating instructions by reading said records one by one (602 - 608, 1002 - 1007).

24. A computer software product according to Claim 15, said medium having further stored thereon:

a code section for storing said transformed program in at least one IC card, wherein said program is loaded in memory by generating an absolute address format program by sequentially relocating the instructions by reading said

records one by one from said IC card.

FIG. 1

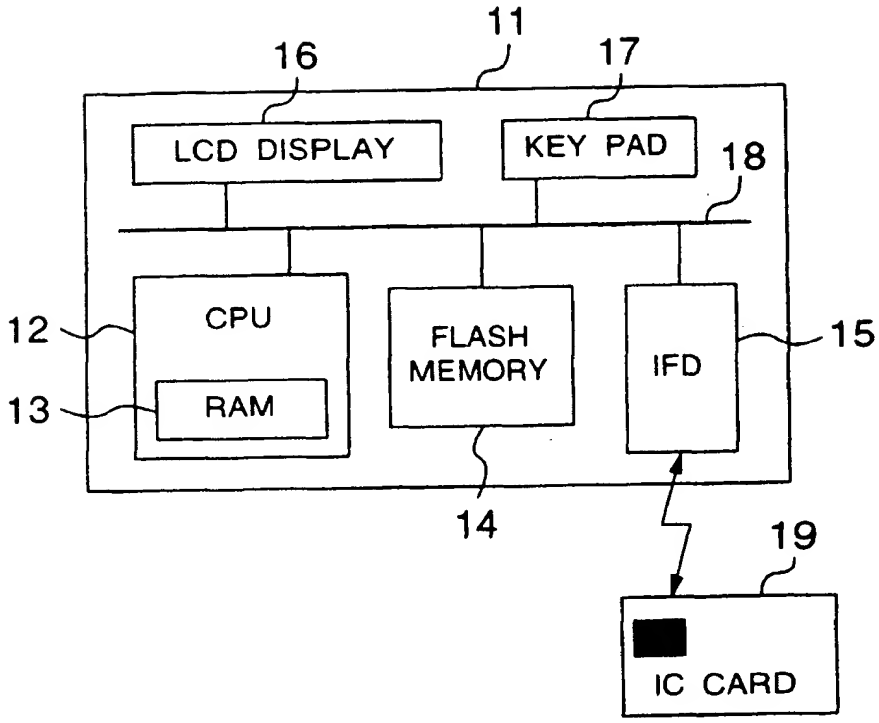


FIG. 2

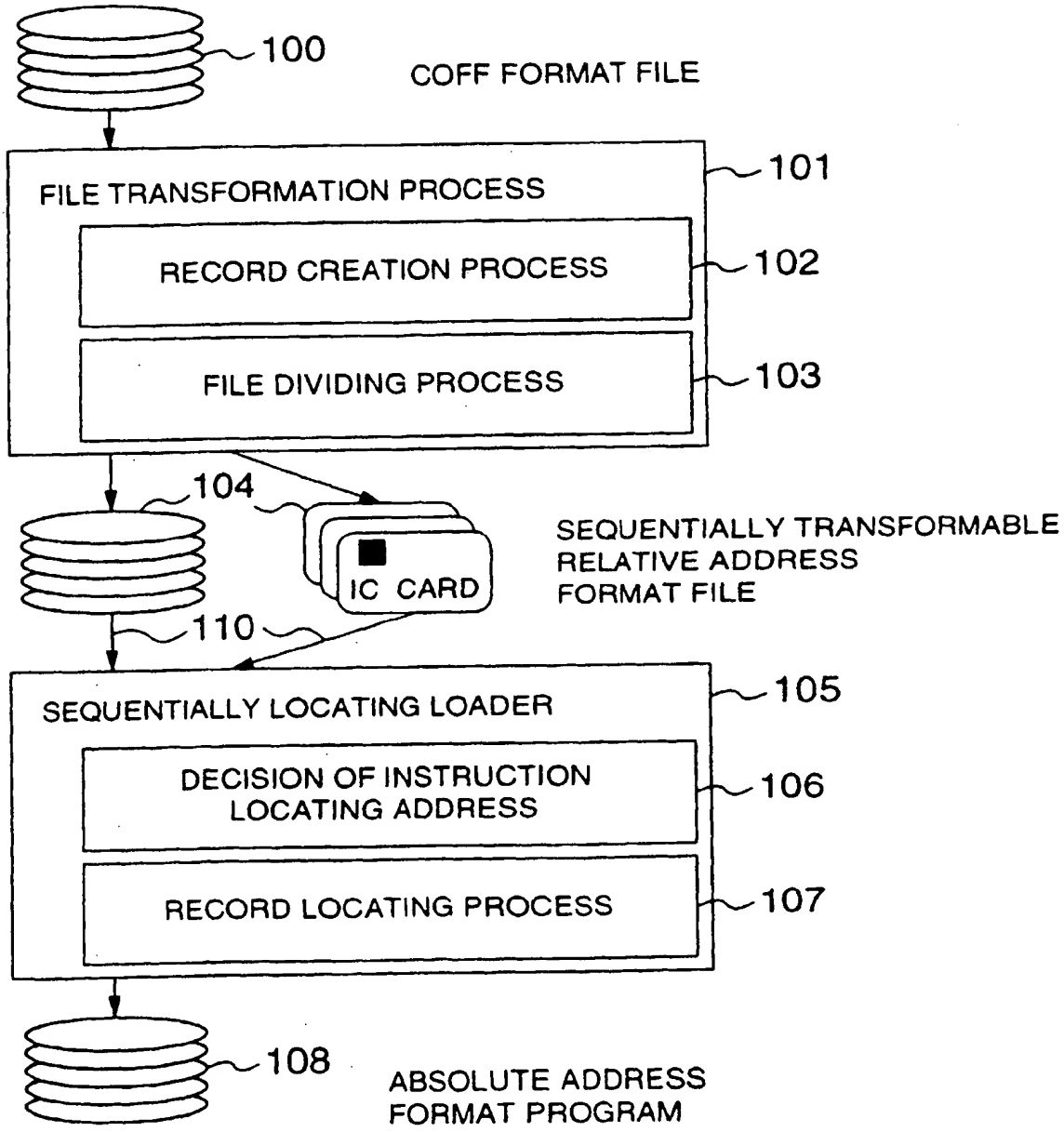

100
COFF FORMAT FILE
FILE TRANSFORMATION PROCESS
101
RECORD CREATION PROCESS
102
FILE DIVIDING PROCESS
103
104
IC CARD
SEQUENTIALLY TRANSFORMABLE
RELATIVE ADDRESS
FORMAT FILE
110
SEQUENTIALLY LOCATING LOADER
105
DECISION OF INSTRUCTION
LOCATING ADDRESS
106
RECORD LOCATING PROCESS
107
108
ABSOLUTE ADDRESS
FORMAT PROGRAM

FIG. 3

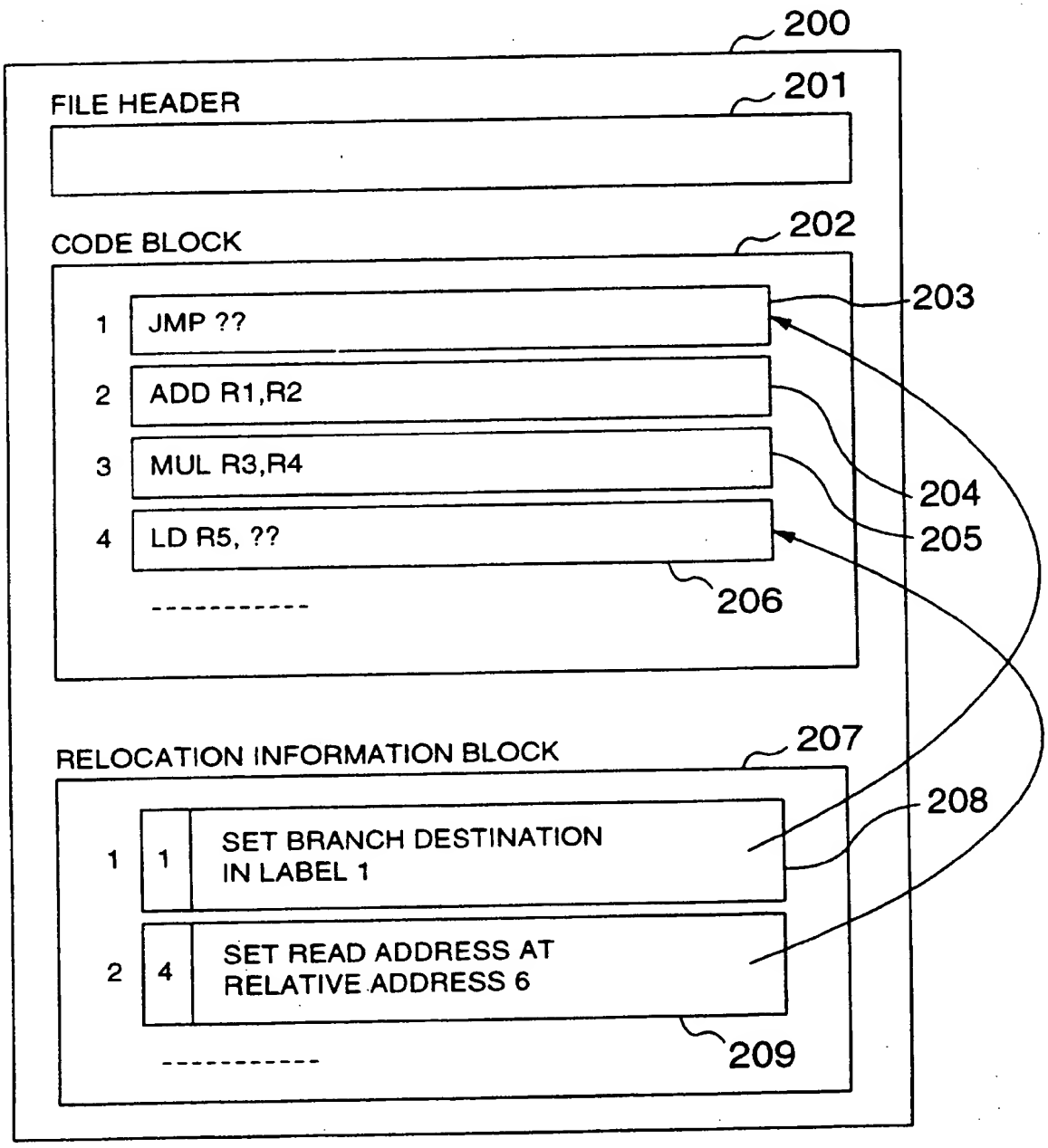

200
FILE HEADER
201
CODE BLOCK
202
1 JMP ??
2 ADD R1,R2
3 MUL R3,R4
4 LD R5, ??
203
204
205
206
RELOCATION INFORMATION BLOCK
207
1 1 SET BRANCH DESTINATION IN LABEL 1
2 4 SET READ ADDRESS AT RELATIVE ADDRESS 6
208
209

FIG. 4

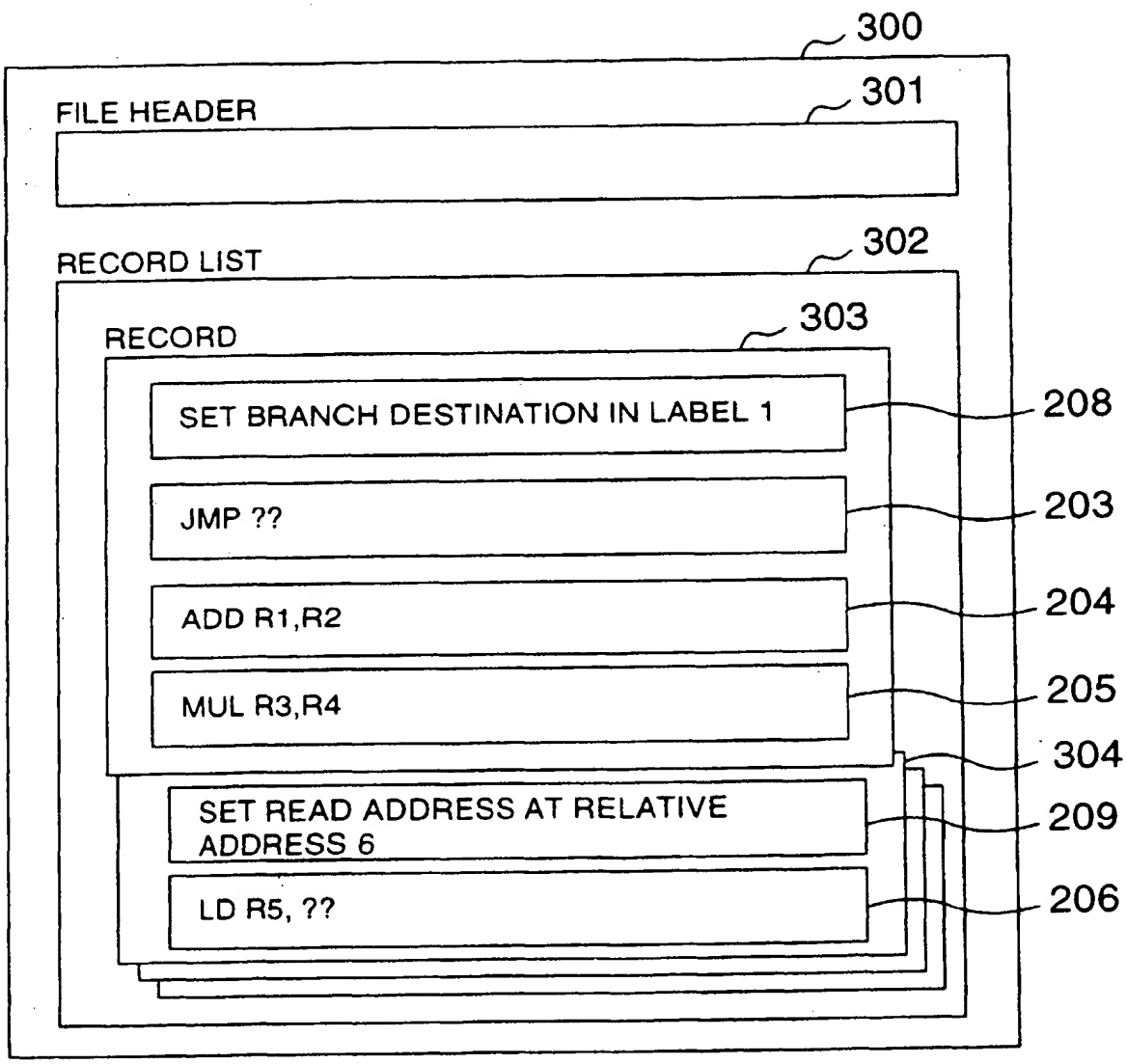

300
FILE HEADER
301
RECORD LIST
302
RECORD
303
SET BRANCH DESTINATION IN LABEL 1
208
JMP ??
203
ADD R1,R2
204
MUL R3,R4
205
304
SET READ ADDRESS AT RELATIVE ADDRESS 6
209
LD R5, ??
206

FIG. 5

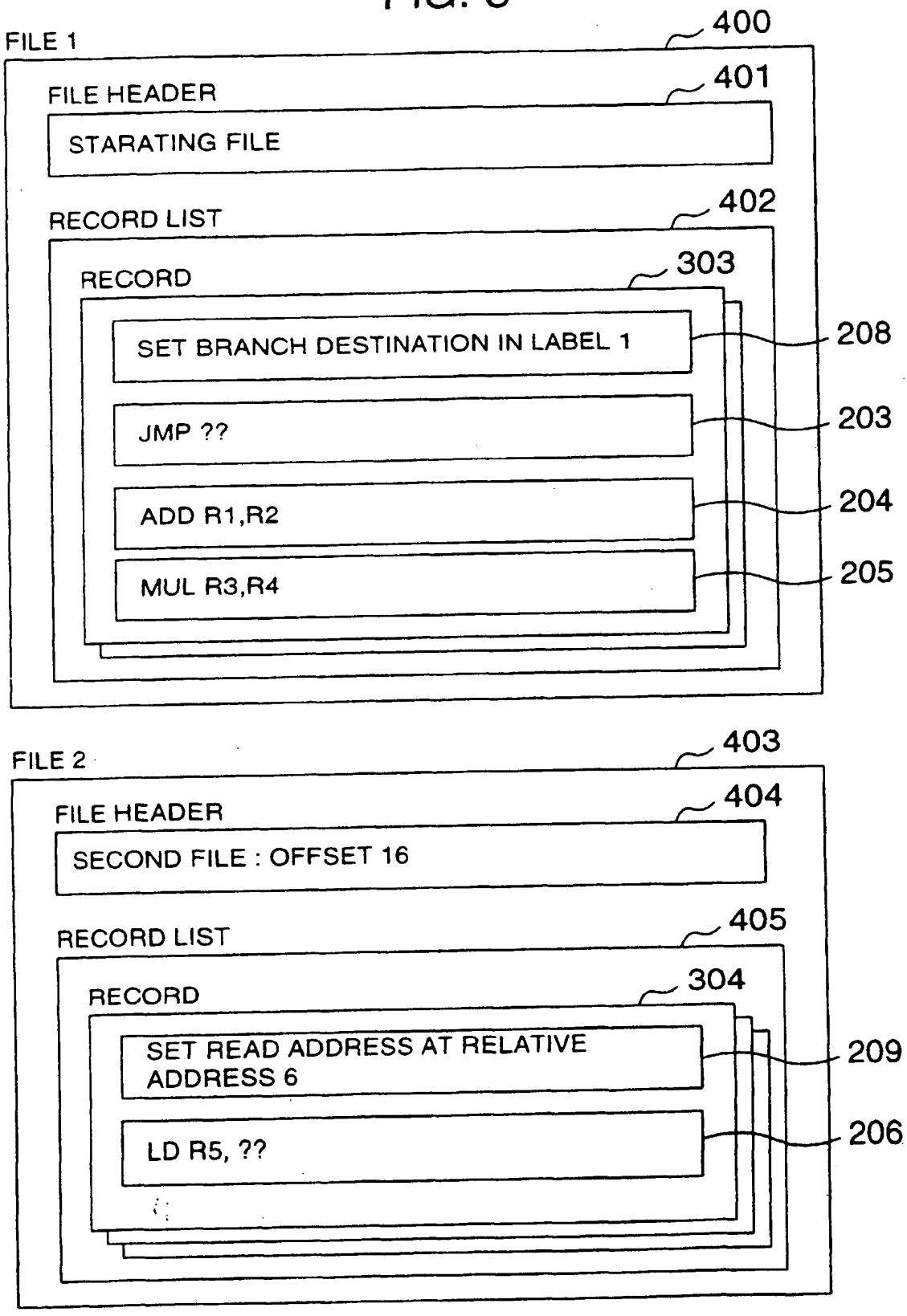

FILE 1
400
FILE HEADER
401
STARATING FILE
RECORD LIST
402
RECORD
303
SET BRANCH DESTINATION IN LABEL 1
208
JMP ??
203
ADD R1,R2
204
MUL R3,R4
205
FILE 2
403
FILE HEADER
404
SECOND FILE : OFFSET 16
RECORD LIST
405
RECORD
304
SET READ ADDRESS AT RELATIVE ADDRESS 6
209
LD R5, ??
206

FIG. 6

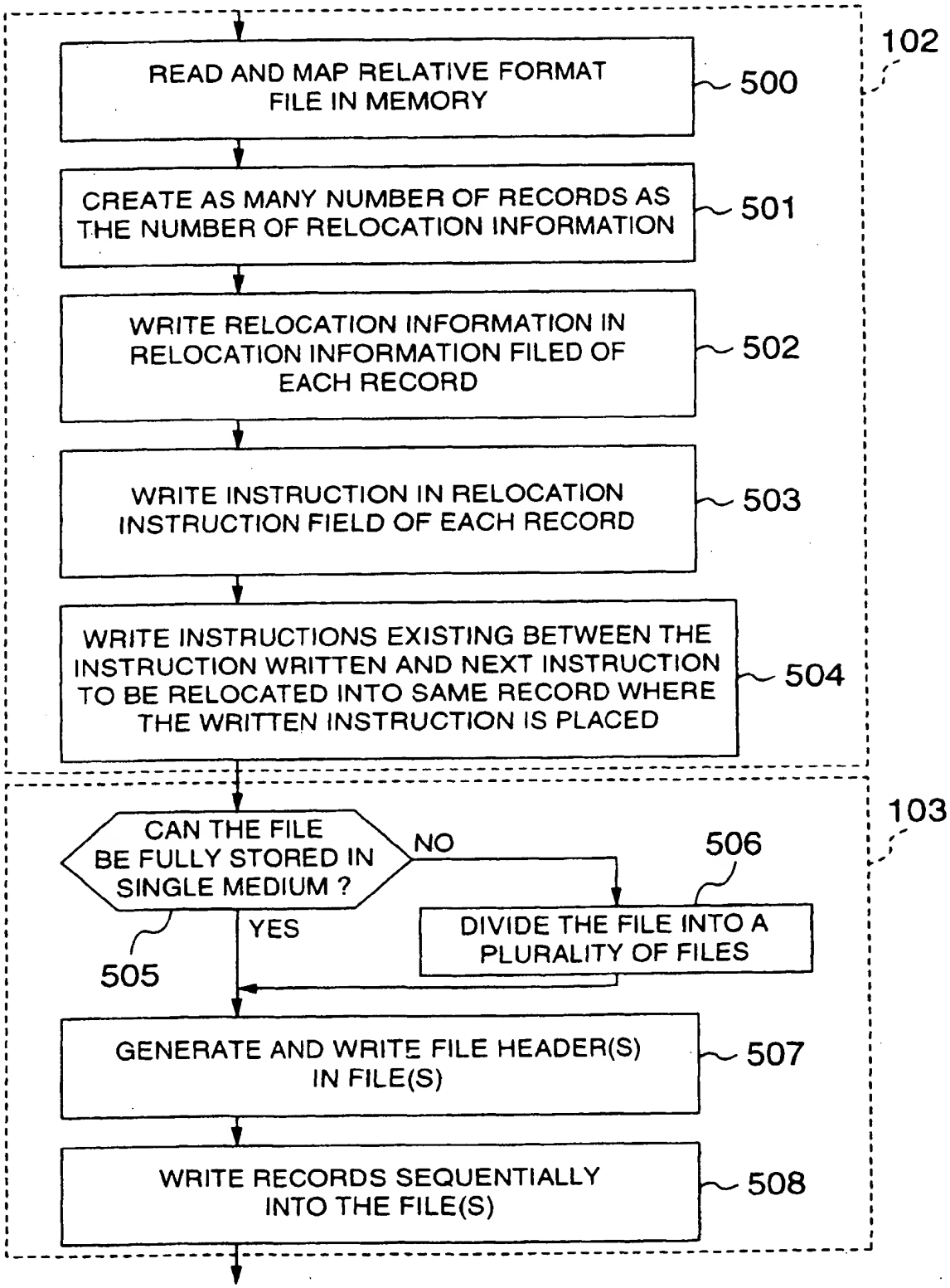


FIG. 7

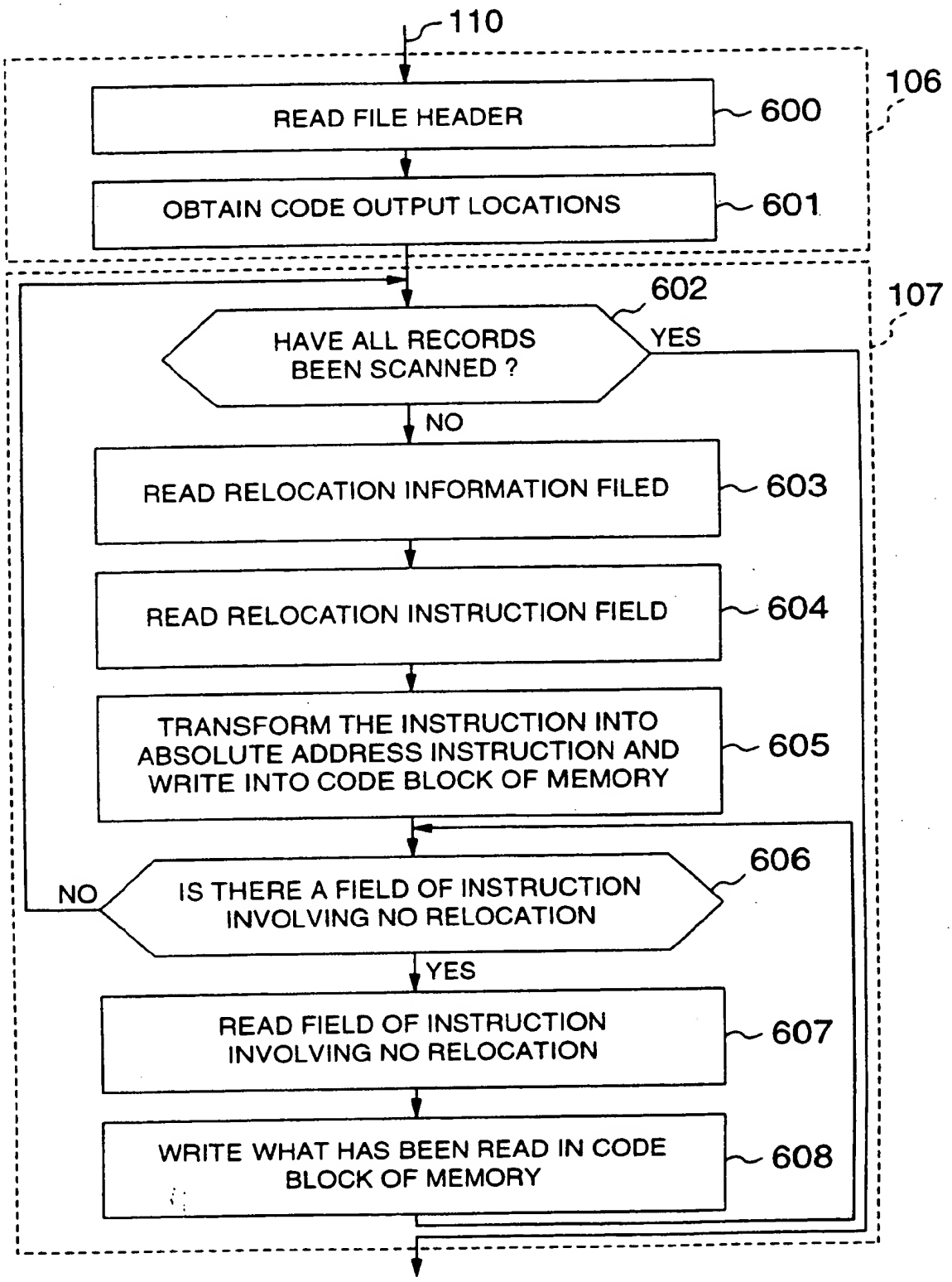


FIG. 8

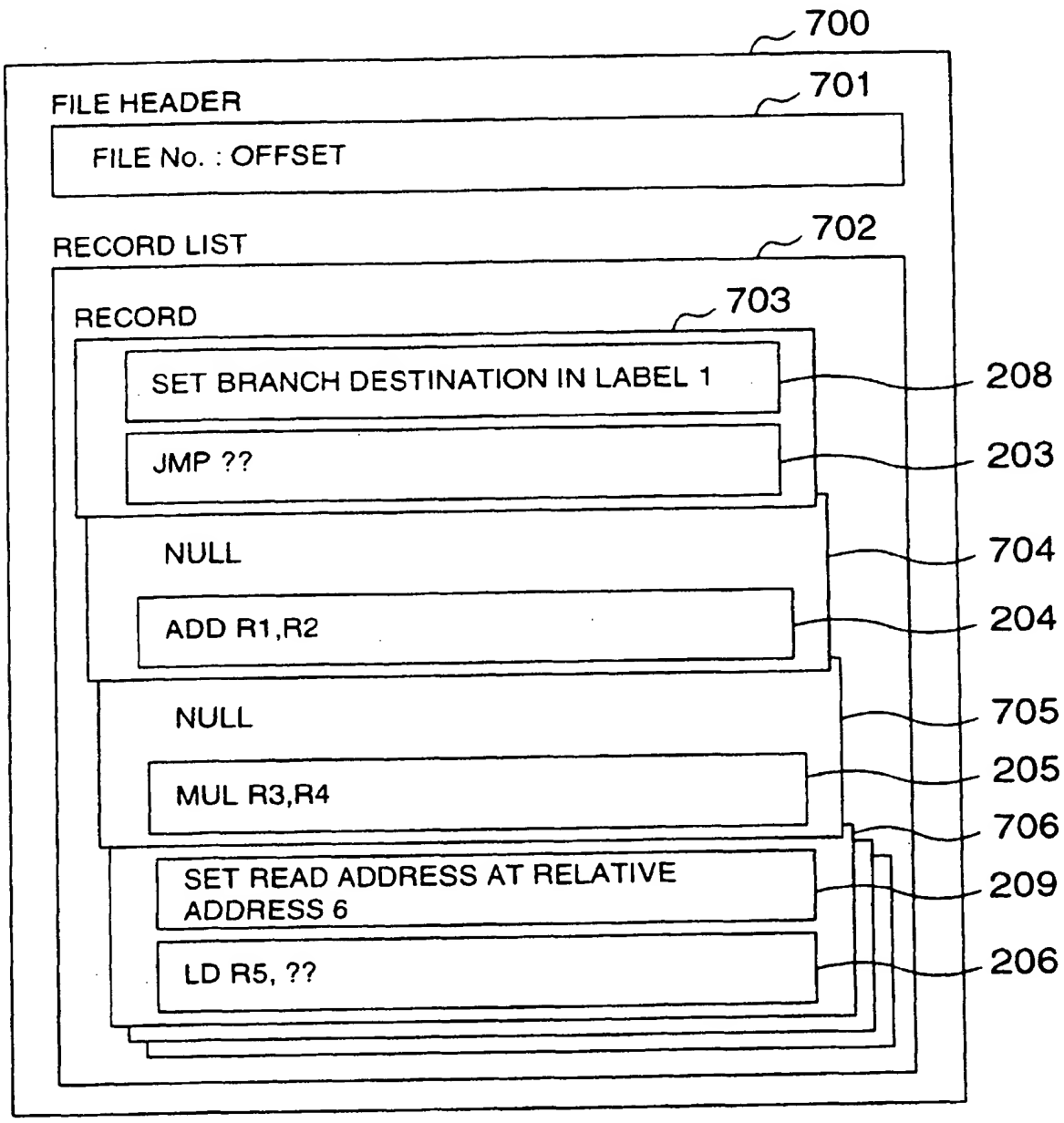

700
FILE HEADER
701
FILE No. : OFFSET
RECORD LIST
702
RECORD
703
SET BRANCH DESTINATION IN LABEL 1
208
JMP ??
203
NULL
704
ADD R1,R2
204
NULL
705
MUL R3,R4
205
706
SET READ ADDRESS AT RELATIVE ADDRESS 6
209
LD R5, ??
206

FIG. 9

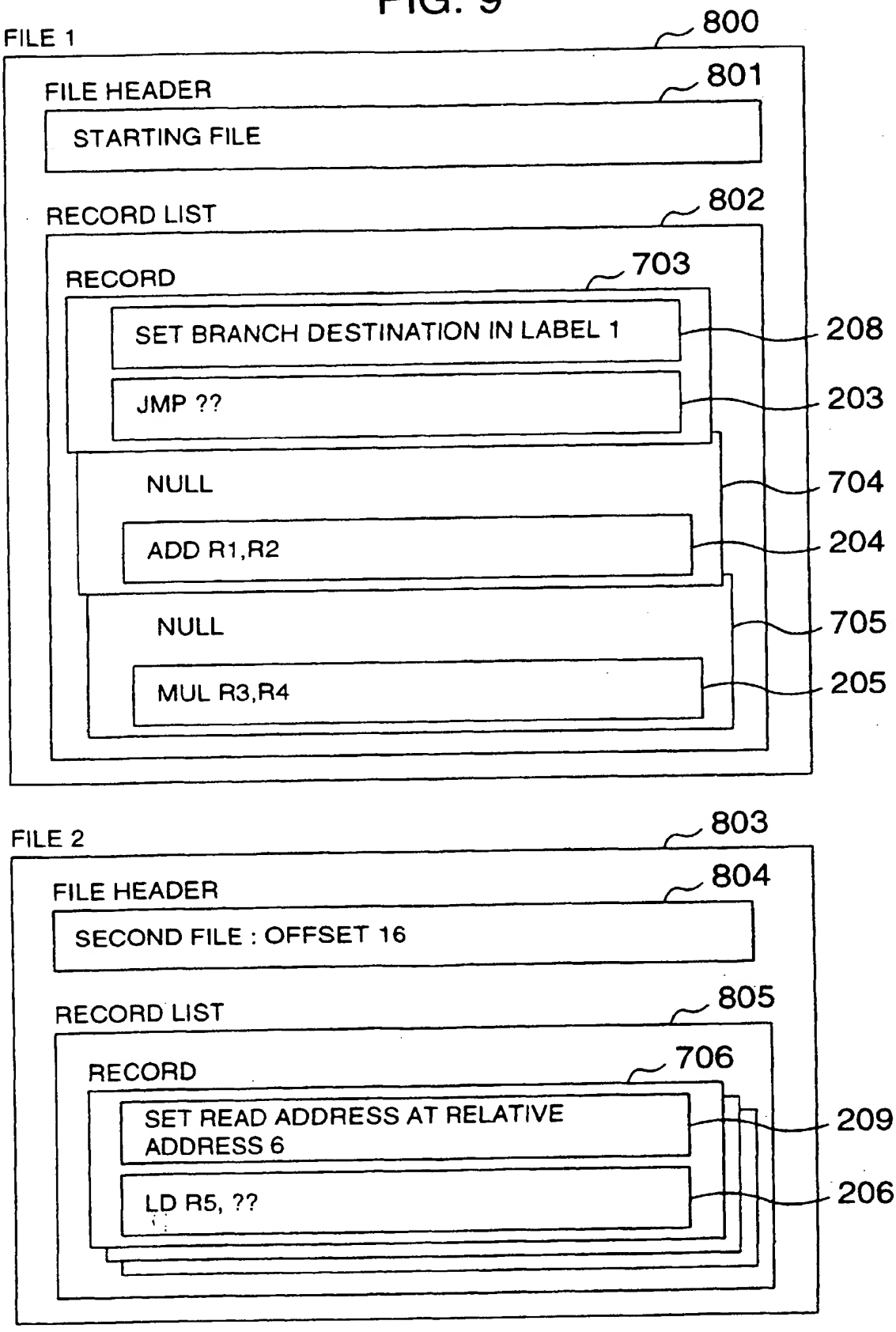

FILE 1
800
FILE HEADER
801
STARTING FILE
RECORD LIST
802
RECORD
703
SET BRANCH DESTINATION IN LABEL 1
208
JMP ??
203
NULL
704
ADD R1,R2
204
NULL
705
MUL R3,R4
205
FILE 2
803
FILE HEADER
804
SECOND FILE : OFFSET 16
RECORD LIST
805
RECORD
706
SET READ ADDRESS AT RELATIVE ADDRESS 6
209
LD R5, ??
206

FIG. 10

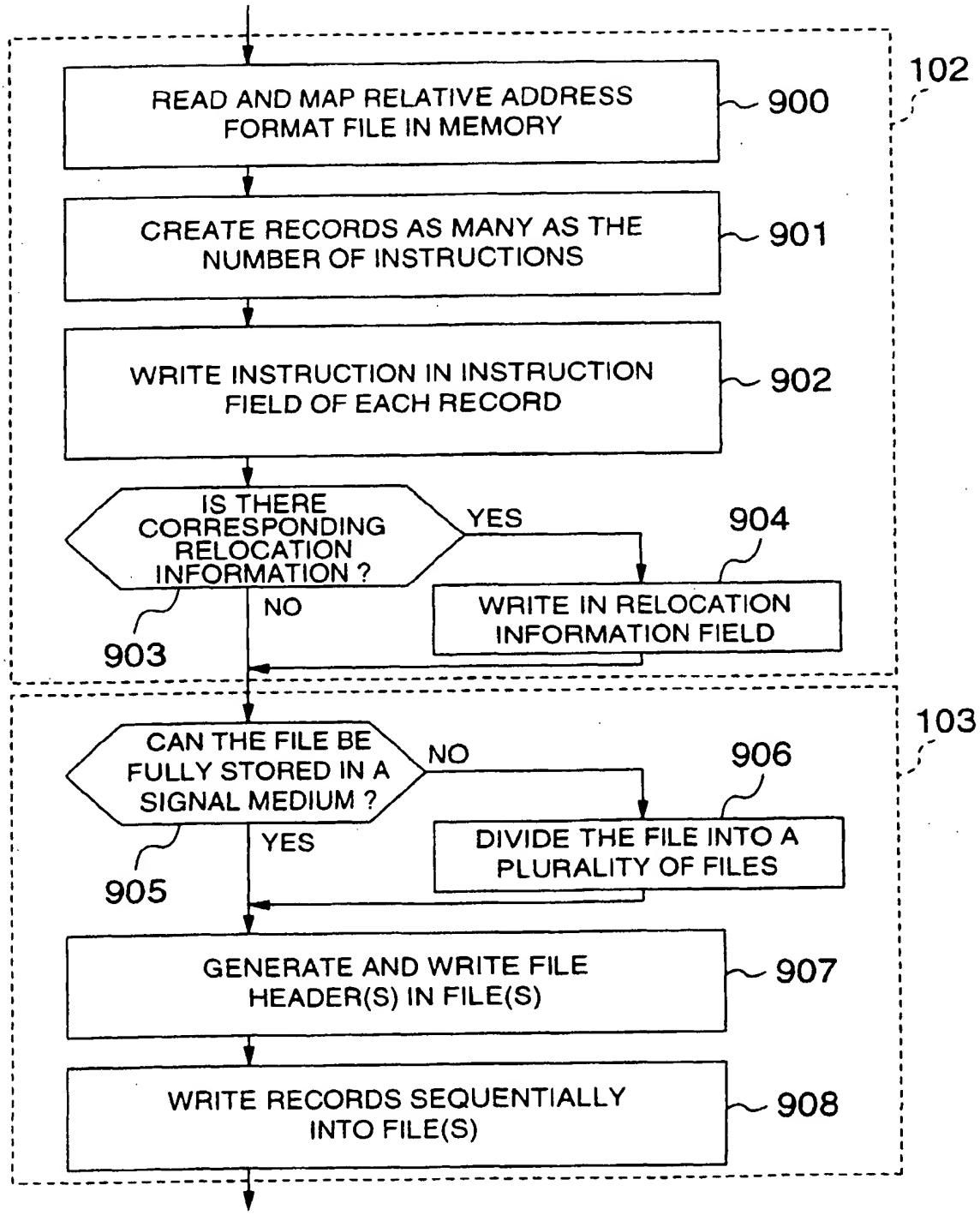

102
READ AND MAP RELATIVE ADDRESS FORMAT FILE IN MEMORY
900
CREATE RECORDS AS MANY AS THE NUMBER OF INSTRUCTIONS
901
WRITE INSTRUCTION IN INSTRUCTION FIELD OF EACH RECORD
902
IS THERE CORRESPONDING RELOCATION INFORMATION ?
YES
904
WRITE IN RELOCATION INFORMATION FIELD
NO
903
103
CAN THE FILE BE FULLY STORED IN A SIGNAL MEDIUM ?
NO
906
DIVIDE THE FILE INTO A PLURALITY OF FILES
YES
905
GENERATE AND WRITE FILE HEADER(S) IN FILE(S)
907
WRITE RECORDS SEQUENTIALLY INTO FILE(S)
908

FIG. 11

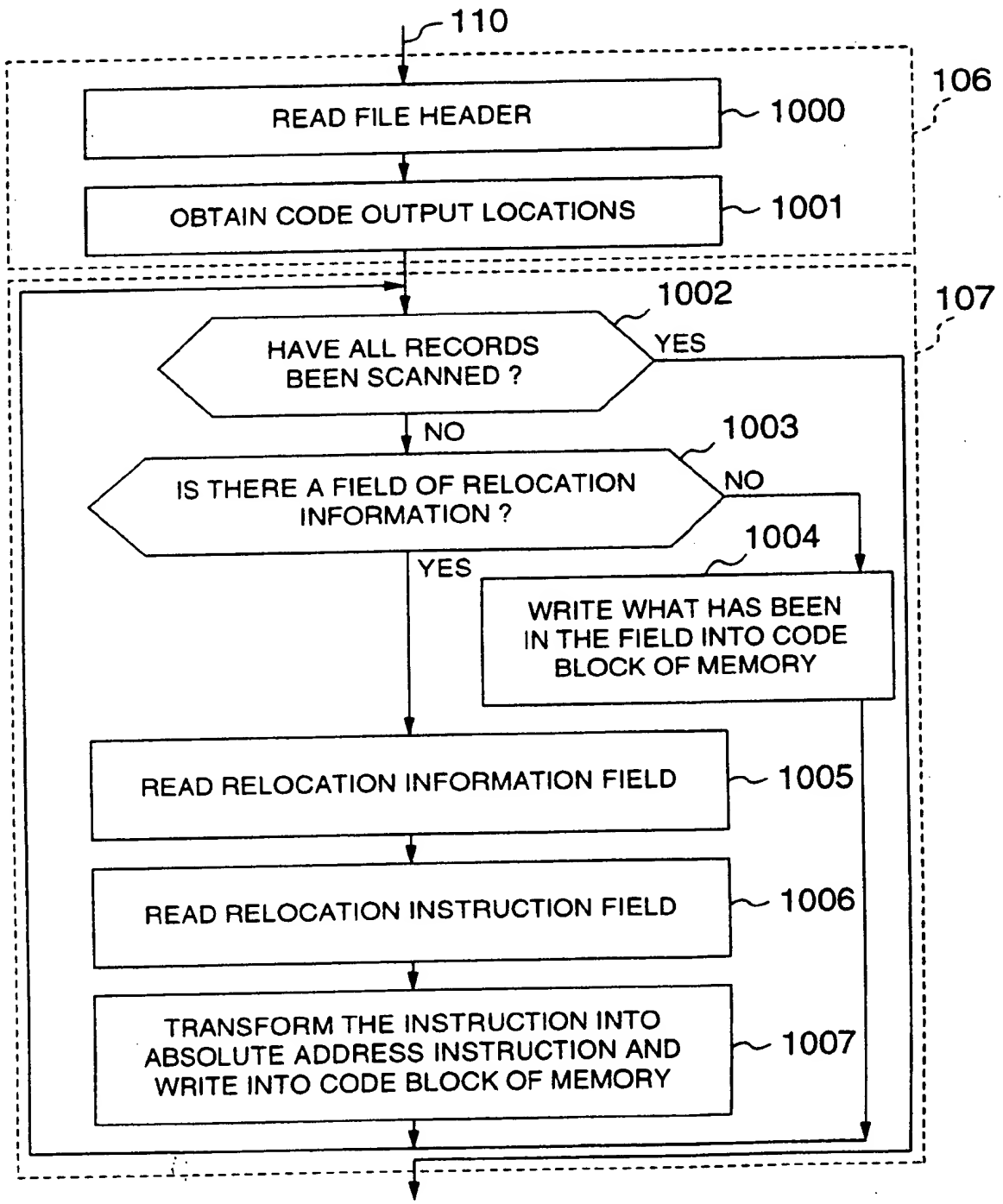

110
106
READ FILE HEADER
1000
OBTAIN CODE OUTPUT LOCATIONS
1001
107
1002
HAVE ALL RECORDS BEEN SCANNED ?
YES
NO
1003
IS THERE A FIELD OF RELOCATION INFORMATION ?
NO
1004
YES
WRITE WHAT HAS BEEN IN THE FIELD INTO CODE BLOCK OF MEMORY
READ RELOCATION INFORMATION FIELD
1005
READ RELOCATION INSTRUCTION FIELD
1006
TRANSFORM THE INSTRUCTION INTO ABSOLUTE ADDRESS INSTRUCTION AND WRITE INTO CODE BLOCK OF MEMORY
1007

FIG. 12

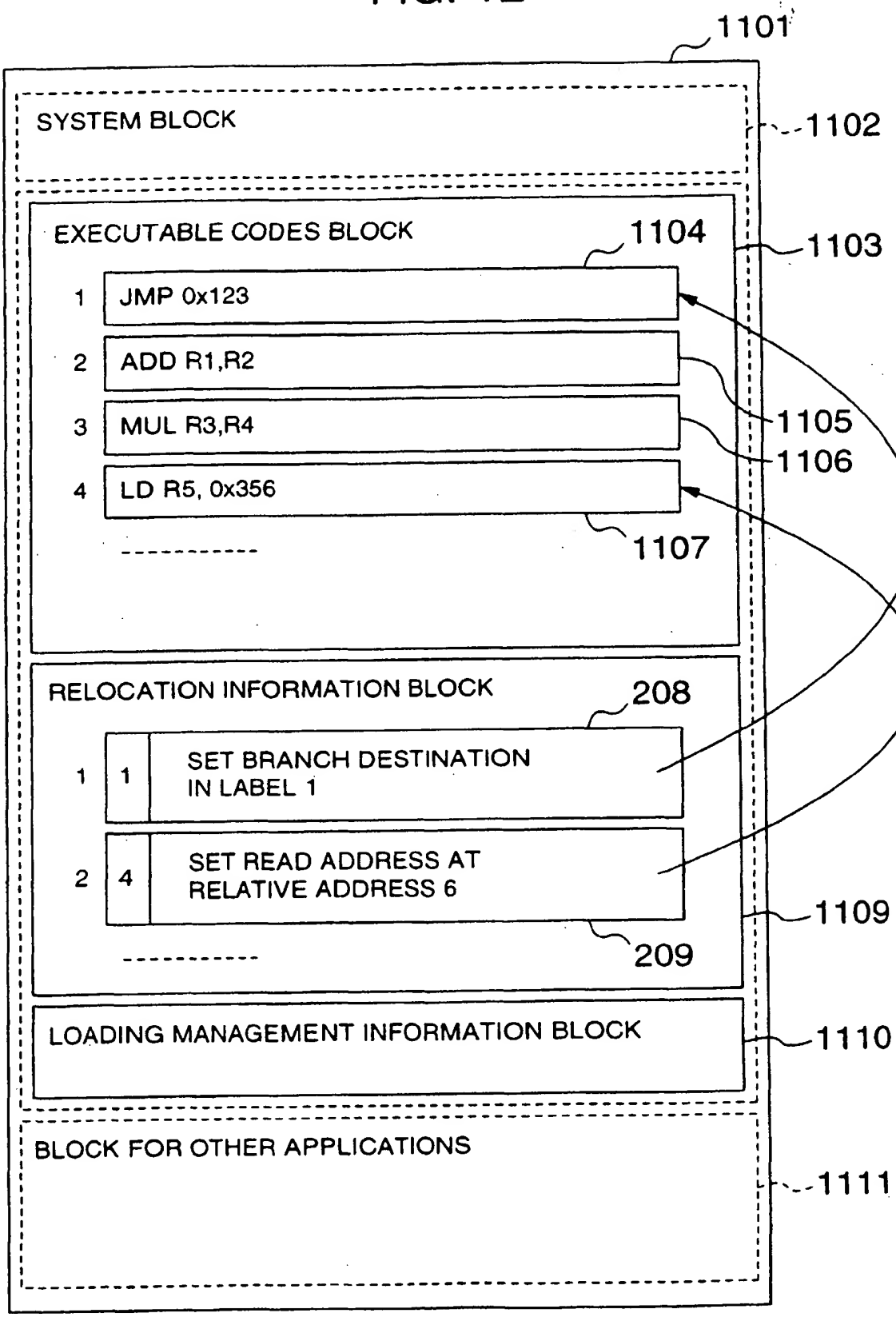

1101
SYSTEM BLOCK
1102
EXECUTABLE CODES BLOCK
1104
1103
1 JMP 0x123
2 ADD R1,R2
3 MUL R3,R4
4 LD R5, 0x356
1105
1106
1107
RELOCATION INFORMATION BLOCK
208
1 1 SET BRANCH DESTINATION IN LABEL 1
2 4 SET READ ADDRESS AT RELATIVE ADDRESS 6
1109
209
LOADING MANAGEMENT INFORMATION BLOCK
1110
BLOCK FOR OTHER APPLICATIONS
1111

FIG. 13

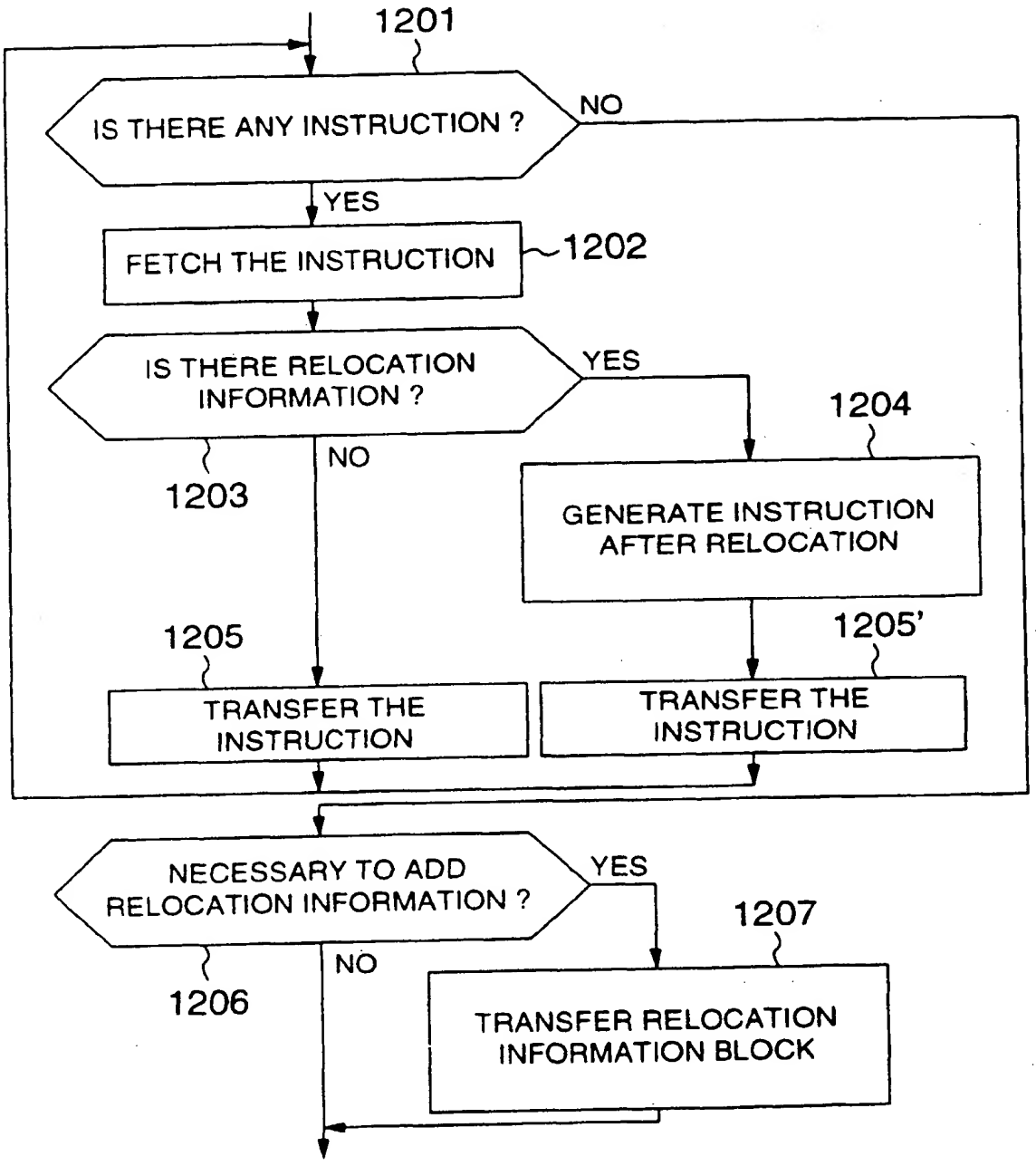

1201
IS THERE ANY INSTRUCTION ?
NO
YES
FETCH THE INSTRUCTION
1202
IS THERE RELOCATION INFORMATION ?
YES
1204
NO
1203
GENERATE INSTRUCTION AFTER RELOCATION
1205
1205'
TRANSFER THE INSTRUCTION
TRANSFER THE INSTRUCTION
NECESSARY TO ADD RELOCATION INFORMATION ?
YES
1207
NO
1206
TRANSFER RELOCATION INFORMATION BLOCK

FIG. 14

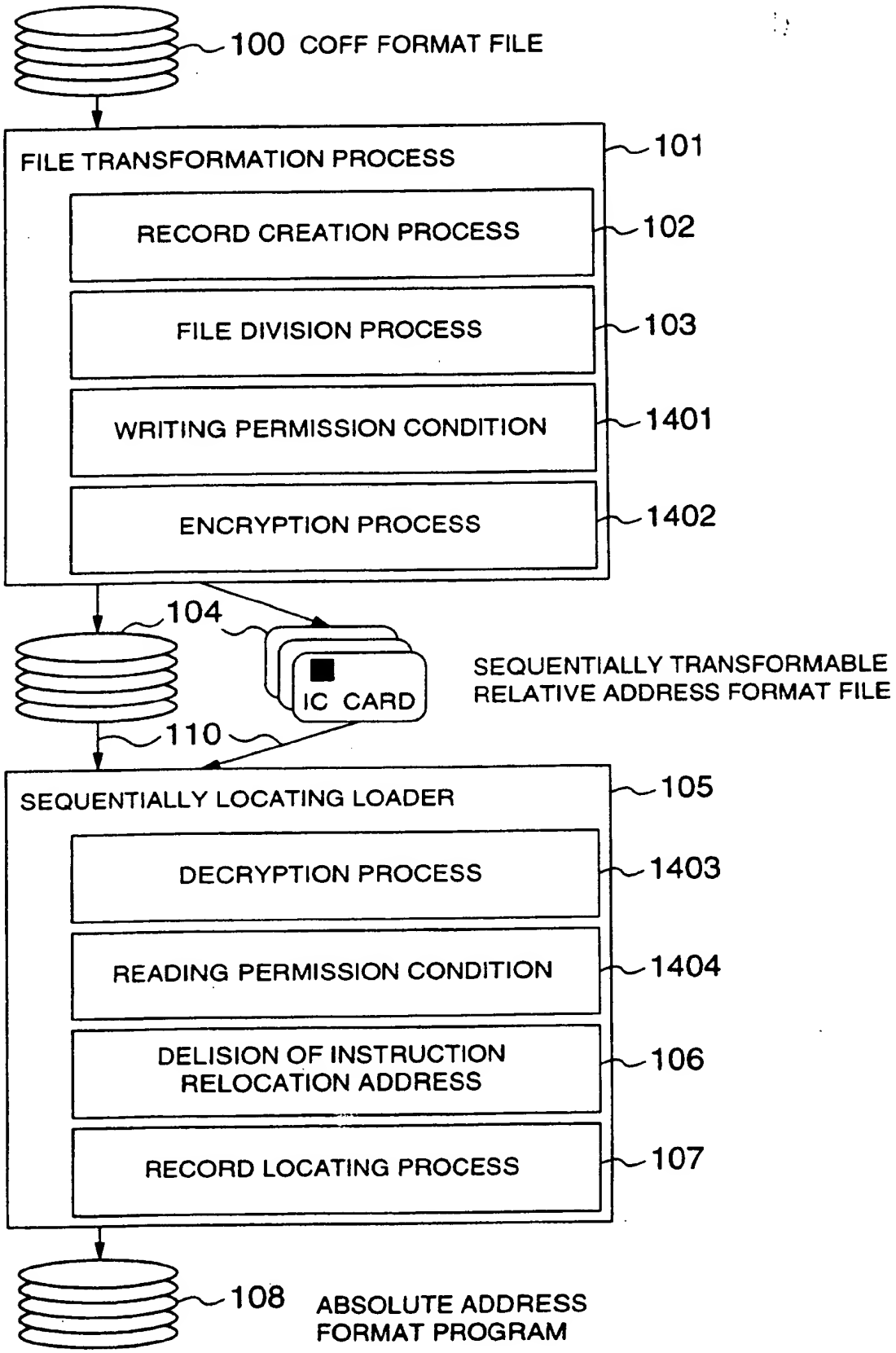
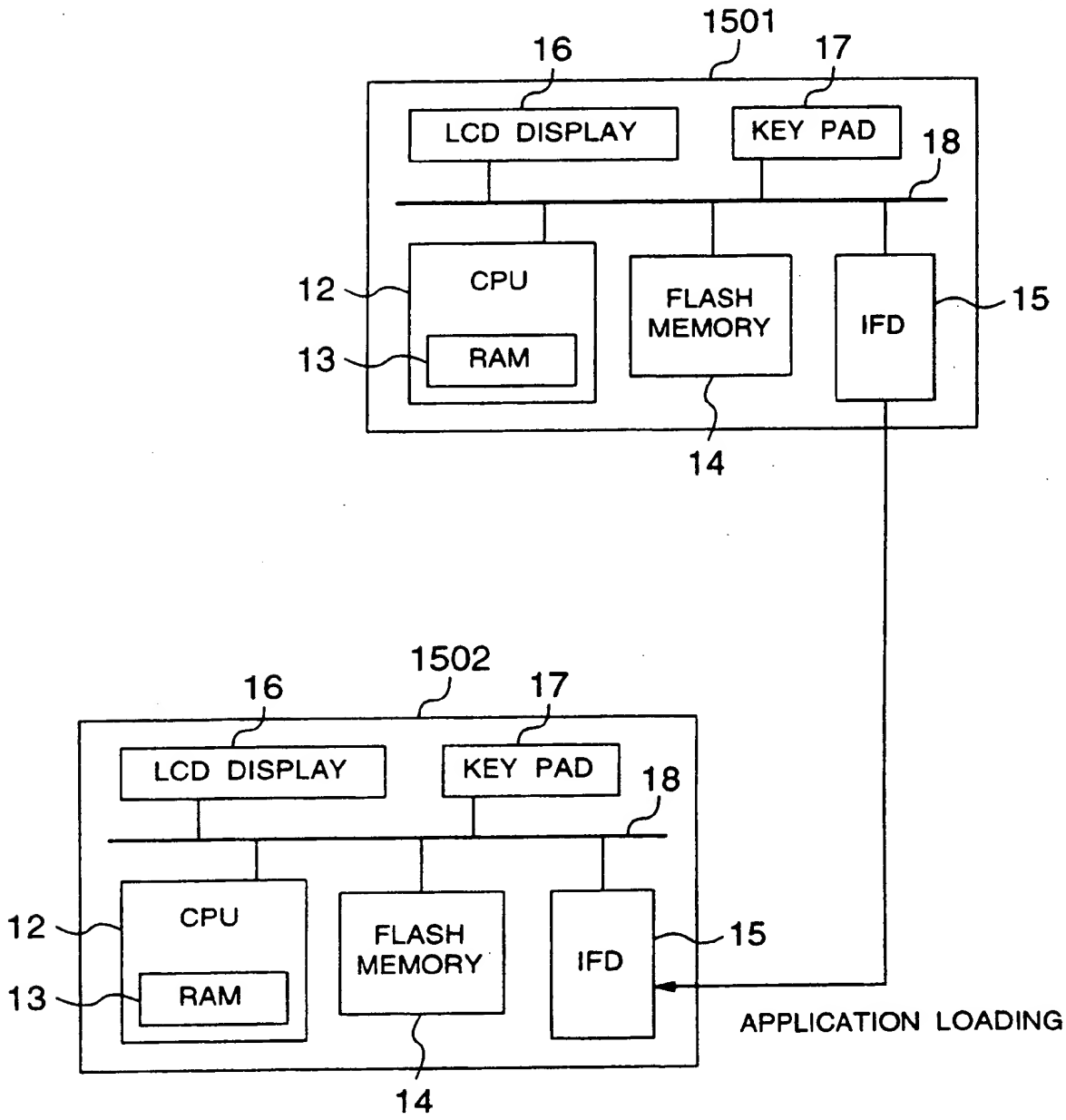

100 COFF FORMAT FILE
FILE TRANSFORMATION PROCESS
101
RECORD CREATION PROCESS
102
FILE DIVISION PROCESS
103
WRITING PERMISSION CONDITION
1401
ENCRYPTION PROCESS
1402
104
IC CARD
SEQUENTIALLY TRANSFORMABLE
RELATIVE ADDRESS FORMAT FILE
110
SEQUENTIALLY LOCATING LOADER
105
DECRYPTION PROCESS
1403
READING PERMISSION CONDITION
1404
DELISION OF INSTRUCTION
RELOCATION ADDRESS
106
RECORD LOCATING PROCESS
107
108
ABSOLUTE ADDRESS
FORMAT PROGRAM

FIG. 15


1501
16
17
LCD DISPLAY
KEY PAD
18
12
CPU
13
RAM
FLASH MEMORY
IFD
15
14
1502
16
17
LCD DISPLAY
KEY PAD
18
12
CPU
13
RAM
FLASH MEMORY
IFD
15
14
APPLICATION LOADING

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 98 11 6802

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
X Y	EP 0 321 000 A (NIPPON ELECTRIC CO) 21 June 1989 * abstract *	1,2,13, 14 3-6,10, 11, 15-19, 23,24	G06F9/445 G06F1/00
A	* column 1, line 1 - column 5, line 10 * * column 16, line 8 - column 17, line 11 *	21	
Y	WO 96 27158 A (INTEL CORP) 6 September 1996 * abstract * * page 1, line 1 - page 3, line 19; claims 1-3 *	3-6,10, 11, 15-19, 23,24	
X A	EP 0 489 204 A (HEWLETT PACKARD LTD) 10 June 1992 * abstract * * column 7, line 16 - column 10, line 23; figures 2-4 *	8,20 2,4,6,7, 9,10,14, 15,17, 19-22	TECHNICAL FIELDS SEARCHED (Int.Cl.6) G06F G07F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	16 December 1998	Kingma, Y

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 98 11 6802

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

16-12-1998

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0321000 A	21-06-1989	JP 1161534 A	26-06-1989
		JP 1177128 A	13-07-1989
		US 5182807 A	26-01-1993
WO 9627158 A	06-09-1996	US 5581768 A	03-12-1996
		AU 4986096 A	18-09-1996
		DE 19681256 T	12-02-1998
		GB 2314182 A	17-12-1997
EP 0489204 A	10-06-1992	DE 69021732 D	21-09-1995
		DE 69021732 T	18-01-1996
		JP 4290127 A	14-10-1992

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82